Posgrado en Optimización

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

División de Ciencias Básicas e Ingeniería

Problema de ruteo del autobús escolar con recolección mixta

Tesis para obtener el grado de

Maestro en Optimización
por
Ing. Eduardo Rojas Silva

Asesores:
Dr. Rafael López Bracho
Departamento de Sistemas, UAM Azcapotzalco

Dr. Javier Ramírez Rodríguez
Departamento de Sistemas, UAM Azcapotzalco

9 de abril de 2015

Dedicatoria

A todas las personas que hicieron que este trabajo fuera posible.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) y a la Universidad Autónoma Metropolitana (UAM) por haber otorgado el financiamiento para la realización de este proyecto de investigación.

A mis asesores el Dr. Rafael López Bracho y el Dr. Javier Ramírez Rodríguez por la confianza depositada en mi, su constante apoyo y tiempo dedicado para la realización de este proyecto de investigación.

A Alberto Rojas y Martha Estela Silva, mis padres, por confiar que las desiciones que tomo siempre son las acertadas.

A mis amigos, Jonathan Nacazona, José Antonio Pérez, Juan Ramón Blanno, Edgar Ivan Hernández, Ana Martínez y Sonia Gabriela Hernández, por su apoyo incondicional.

A todos los profesores y amigos del posgrado en optimización por todo el conocimiento que compartieron conmigo.

Resumen

En este trabajo se presenta una variante del "Problema de ruteo del autobús escolar"(SBRP) clásico, en el cual se trata de minimizar la distancia que se recorre en cada ruta. Concretamente el problema propuesto es una variante del problema de ruteo del autobús escolar para estudiantes con necesidades especiales, en el cual se considera la existencia de dos tipos de de estudiantes, con atributos diferentes al momento de decidir en donde se van a recoger, a diferencia de otras variantes existentes en la literatura, se considera la parte de asignación de los estudiantes como parte del problema ya que se considera que este parámetro influye al momento en que se crean las rutas, aun cuando la complejidad del problema se vea incrementada.

La idea principal de este trabajo consiste en explorar tanto los métodos heurísticos como los exactos, para poder obtener una solución a la variante propuesta, ya sea de forma exacta o aproximada. De tal manera que se pueda identificar en qué casos conviene usar uno de estos métodos.

El problema se modela como un programa lineal entero mixto, el cual posteriormente se resuelve mediate el software glpk y se encuentran soluciones para instancias pequeñas, que se toman como base para compararlas con las encontradas con el método heurístico utilizado.

Para la parte heurística se optó por utilizar una de las técnicas clásicas: Un algoritmo glotón aleatorizado y adaptativo (GRASP, del inglés Greedy Randomized Adaptive Search Procedures), dado que se parte de la formulación matemática previamente obtenida para la parte exacta, fácilmente se puede adaptar para la elaboración del GRASP. La técnica GRASP es una meta-heurística muy rápida que genera soluciones de alta calidad, la cual se divide en dos fases: una de construcción de soluciones, que utiliza para ello un algoritmo glotón aleatorizado y una fase de mejora, la cual realiza una búsqueda local. Este proceso se repite varias veces y la mejor solución encontrada se toma como el resultado, lo que al final nos proporciona una solución que si bien no garantiza que sea la óptima, sí es de buena calidad y se puede obtener en un tiempo mucho menor en comparación con el tiempo que tarda encontrar la solución con un método exacto, cuando éste es capaz de encontrar una.

Finalmente nos enfocamos en la versión más simple y general del problema para dar un

algoritmo de aproximación, el cual se basa en un algoritmo que originalmente se diseñó para resolver una variante del problema de ruteo vehícular.

Contenido

Lista de Figuras

Lista de Tablas

Lista de algoritmos

Capítulo 1
Introducción

Uno de los problemas al que se enfrentan la mayoría de las grandes ciudades, es el tráfico que se genera en las llamadas horas pico. Mucho de este tráfico es generado por los padres de familia los cuales llevan a sus hijos a las escuelas. A raíz de esto muchos países han optado por implementar un sistema de transporte escolar que tiene por objetivo reducir la cantidad de vehículos que circulan por las calles en las horas pico.

Si bien, esto genera un beneficio para la mayoría de la sociedad, también implica un gasto para los administradores de las escuelas. El gasto operacional se traduce en sueldos para los conductores, mantenimiento de los autobuses, costo de la gasolina, seguro contra accidentes, entre otros. Por lo que cualquier pequeño costo que se pueda reducir es de suma importancia.

Esto hecho ha captado la atención de numerosos investigadores los cuales buscan encontrar la manera de reducir los costos de operación de los autobuses sin afectar la calidad del servicio. Esto no es una tarea fácil ya que un área escolar con varias escuelas se traduce en cientos de estudiantes, decenas de autobuses y decenas de posibles paradas de autobús.

Es por eso que muchos de los investigadores en un principio buscaron una manera diferente de representar este problema de forma que visualmente pueda dar una idea de todos los sujetos que están involucrados en el problema, es ahí donde entra la teoría de gráficas, ya que mediante una gráfica fácilmente se pueden representar todos los elementos involucrados, estudiantes, paraderos y escuela pueden representarse mediante vértices y las aristas representan la trayectoria que siguen los autobuses para recoger a los estudiantes.

Si bien, el hecho de tener representado el problema de forma gráfica puede ayudar a obtener ideas de cómo se puede resolver, tratar de resolverlo directamente a través de su representación gráfica no es una tarea sencilla y menos para instancias grandes del problema, es por eso que se buscaron otras opciones para representar el problema, siendo la más utilizada mediante una formulación matemática.

En general el problema de ruteo del autobús escolar (SBRP[1]), es una formulación matemática del problema cuya solución determina la forma mediante la cual los estudiantes serán transportados a la escuela desde su casa o viceversa, de manera segura, económica y conveniente. El modelo incluye métodos para seleccionar las paradas de autobús a utilizar de todos los lugares posibles, la asignación de los estudiantes a las paradas de autobús, asi como la generación de las rutas de los autobuses.

Este problema es una extensión más compleja del problema de ruteo vehicular (VRP[2]), ya que el hecho de considerar estudiantes, en vez de productos a entregar, implica el tener una serie de restricciones enfocadas en la calidad del viaje de los estudiantes.

1.1. Motivación

La Ciudad de México es una de las ciudades con un alto parque vehicular con respecto a su tamaño. De acuerdo con datos de la Secretaria de Transportes y Vialidad entre 1996 y 2003 los vehículos que circularon por la principales vialidades del D.F. se incrementaron entre un 70 % y un 140 %. Donde los vehículos particulares ocupaban el 90 % de las vialidades de la ciudad, lo que ocasiona severos congestionamientos viales y una disminución en la velocidad de circulación.

De acuerdo con el estudio llamado *"Mexico City Vehicle Activity Study"* [8], la velocidad promedio de circulación en el 2004 era de 21 km/hr, actualmente la velocidad promedio es de 12 km/hr. Si bien se ha tratado de solucionar el problema mejorando la infraestructura vial dentro de la ciudad, estos esfuerzos se han visto rebasados por el incremento en el parque vehicular que hay año con año.

Dicho estudio también muestra el papel que juegan las escuelas con respecto al congestionamiento vial, dado que existen más de 4,200 escuelas públicas de nivel básico en la Ciudad de México y aproximadamente 1,140 escuelas particulares del mismo nivel, se sabe que entre 45 y 50 % de los alumnos llegan a las escuelas en auto particular y que en promedio cada vehículo transporta 1.3 alumnos.

Lo anterior ocasiona que en las horas pico (la hora de entrada y salida de los estudiantes) los vehículos que se destinan exclusivamente a llevar (o recoger) estudiantes, representan entre el 20 y 25 % de los vehículos que circulan en ese momento.

Si bien el gobierno ha tomado medidas para intentar solucionar esta problemática, la ma-

[1]SBRP: School Bus Routing Problem.
[2]VRP: Vehicle Routing Problem.

yoría de ellas están enfocadas a mejorar e incluso ampliar las vialidades, por desgracia estas no han sido del todo efectivas.

En vista de ello se optó por intentar otro enfoque, el cual consiste en implementar un programa de transporte escolar, ya que en otros países la implementación de un sistema parecido ha traído grandes beneficios para sus ciudades, sin embargo, a diferencia de estos países, en México este proyecto se lanzó sin un previo estudio de el entorno y las necesidades que esto implica, lo que puede desencadenar que este programa fracase. Es con base en esta necesidad que se plantea una versión del problema de ruteo del autobús escolar, en el cual se incluyen algunas de las problemáticas que se pueden encontrar en la Ciudad de México.

Capítulo 2
Problemas de ruteo

En las últimas décadas se ha experimentado un incremento en el uso de paquetes de optimización, basados en técnicas de programación matemática e investigación de operaciones, dichos paquetes han mostrado tener un buen desempeño en problemas de asignación de recursos y servicios en diferentes localidades. Para resolver problemáticas reales grandes con respecto a cómo distribuir productos a diferentes localidades, se han utilizado procedimientos computacionales para ayudar a planear la distribución y los resultados han mostrado que se llega a ahorrar entre un 5 % y 20 % en el costo total del transporte en comparación a la forma habitual de realizar la planeación.

El éxito de estos paquetes computacionales, radica en el desarrollo de software y hardware integrados a los sistemas de información que hacen posible el procesar una gran cantidad de datos. Otro factor no menos importante es el desarrollo e implementación de modelos matemáticos y algoritmos, estos modelos toman en consideración todas las características (o al menos las más importantes) de los problemas de distribución de productos que se pueden encontrar en un entorno real, por lo que sus correspondientes algoritmos e implementaciones computacionales proveen resultados para instancias reales en un tiempo relativamente aceptable.

Los problemas de distribución de productos a clientes, así como de proporcionar servicio de transporte a los ciudadanos, dependen mayormente de la cantidad de vehículos disponibles, determinar la forma eficiente de disponer de estos vehículos y la forma de acomodar a la gente, entra en la categoría de problemas de ruteo y de asignación de horarios.

Dada una demanda que hay que satisfacer en diferentes ubicaciones a través de una red de transporte la cual es utilizada por los vehículos, decidir la configuración mediante la cual los vehículos se moverán se clasifica como un problema de ruteo, este tipo de problemas por lo regular requiere algún tipo de espeficación en cuanto a la secuencia de ubicaciones que el vehículo puede visitar, las cuales no tienen restricciones de tiempo ni de precedencia entre ellas. El problema del agente viajero y el problema de ruteo vehícular son dos ejemplos de problemas de ruteo.

Una de las formas de medir la dificultad del problema es mediante la complejidad computacional.

Definición 2.1 *Un problema de decisión se dice polinomial si existe algún algoritmo polinomial que lo resuelva. El conjunto de estos problemas es la clase P.*

Definición 2.2 *Un problema de decisión se dice polinomial no determinista si existe algún algoritmo polinomial no determinista que lo resuelva. El conjunto de estos problemas es la clase NP.*

Definición 2.3 *Una reducción es una transformación de un problema en otro problema.*

Definición 2.4 *Una reducción uno a uno, es una reducción en la cual se convierte una instancia de un problema de decisión C en una instancia de un problema de decisión B.*

Definición 2.5 *Un problema de decisión B es NP-completo si*

(a) $B \in NP$ *y*

(b) cada problema en NP se puede reducir en tiempo polinomial a B.

Definición 2.6 *Un problema B es NP-duro, si todo problema C en NP se puede reducir a B en tiempo polinomial.*

Definición 2.7 *Un problema de NP optimización* $OPT : I \to S$ *cumple que:*

(a) Su conjunto I de instancias se puede reconocer en tiempo polinomial, (es decir, el problema de decisión $e \in I$ *está en P).*

(b) Si $e \in I$ *y* $S(e)$ *es su conjunto de soluciones entonces:*

- *i) Existe un polinomio p tal que si* $s \in S(e)$ *entonces* $|s| \leq p(|e|)$.
- *ii) Si* $|t| \leq p(|e|)$*, se puede puede decidir en tiempo polinomial si* $t \in S(e)$.

(c) La función objetivo $m : I \times S \to \mathbb{Z}$ *se puede calcular en tiempo polinomial.*

La mayoria de los problemas de ruteo son NP-duros, lo que significa que incluso instancias moderadas en tamaño del problema son difíciles de resolver (debido a la cantidad de tiempo de procesamiento que requieren), por lo que la mayoría de los investigadores optan por utilizar heurísticas para encontrar una solución del problema en vez de algún algoritmo determinístico.

2.1. Problema de Ruteo Vehicular

El problema de ruteo vehícular (VRP) consiste en encontrar un conjunto óptimo de rutas, el cual será recorrido por una flotilla de vehículos, los cuales darán servicio a un conjunto de clientes previamente dado. Este es uno de los problemas de optimización combinatoria más importantes.

El VRP fue introducido por primera vez por Dantzig y Ramser [7] en 1959 con el nombre de *"truck dispatching problem"*. En el que describen la aplicación de un caso real, el cual consiste en entregar gasolina a las estaciones de servicio que lo requieran, propusieron la primera formulación de programación matemática, así como un método heurístico cuya solución inicial consistía en que cada vehículo atendía a un sólo cliente.

Algunos años después, en 1964, Clarke y Wright [6] propusieron un heurístico glotón que mejoraba el enfoque de Dantzig-Ramser, ahora comúnmente conocido como *"the saving method"*. Al igual que en el método de Dantzig y Ramser, el algoritmo comienza con una solución inicial donde cada vehículo atiende a un único cliente. Después el método compara las distancias *"savings"* obtenidas al combinar el fin de una ruta con el principio de otra ruta, para todas las combinaciones posibles. Las rutas con las mejores distancias son combinadas y se actualizan las distancias. Esto continúa hasta que las rutas no pueden ser combinadas debido a las restricciones. Este método es simple y fácil de implementar y ha sido la base de varios programas de ruteo.

Después de que estos dos artículos fundamentales fueran publicados, cientos de modelos y algoritmos fueron propuestos para encontrar soluciones óptimas y aproximadas para diferentes versiones del VRP. Las instancias más grandes que pueden resolverse mediante los métodos exactos propuestos contienen a lo más 100 clientes, instancias con un mayor número de clientes sólo pueden resolverse en algunos casos especiales.

El problema de ruteo vehícular clásico consiste en un conjunto de rutas de repartición que utilizan los vehículos, los cuales parten desde un depósito, buscando minimizar la distancia total que recorren.La demanda de cada vértice (cliente), está dada de forma determinística y cada vehículo tiene una capacidad definida. Es fácil observar que el problema del agente viajero, es un caso especial del problema de ruteo vehícular cuando el vehículo tiene una capacidad infinita, por lo que el VRP es *NP-duro*. En el ejemplo de la fígura 2.1 la demanda de los vértices 3, 4, 5 y 6 es de uno, la demanda de los vértices 1 y 2 es de dos y la capacidad de los vehículos es de cuatro.

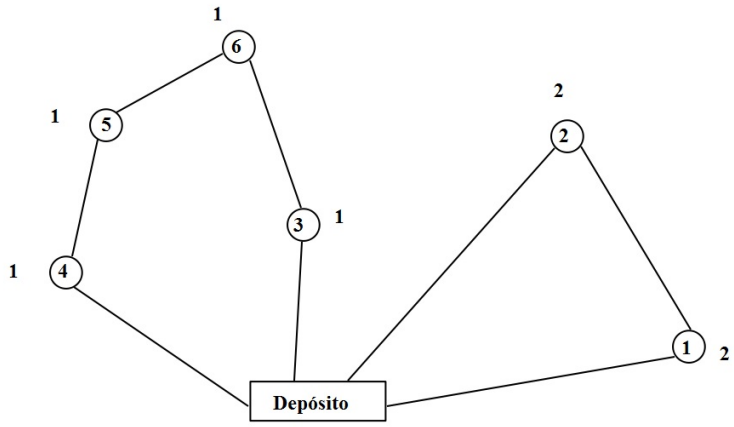

Figura 2.1: Ejemplo de VRP

En las siguientes subsecciones revisaremos algunas de las formulaciones y algunas heurísticas importantes, así como métodos exactos para el VRP.

Para las siguientes subsecciones consideraremos que el VRP tiene las siguientes características, n clientes, el depósito tiene el índice 0 y se consideran K vehículos, d_i denota la demanda solicitada por el cliente i y C denota la capacidad que tienen los vehículos.

2.1.1. Descripción del problema y formulaciones

El problema de ruteo vehicular (VRP) de costo mínimo consiste en encontrar una colección de K rutas vehiculares tal que:

1. Cada vehículo recorre exactamente una sola ruta.
2. Cada ruta inicia en el depósito y regresa al depósito.
3. Cada cliente es visitado por un solo vehículo.
4. El tamaño total de las entregas que realiza cada vehículo a los clientes, no puede superar la capacidad del vehículo.

Para el VRP se asumen diferentes características para los vehículos, clientes y objetivos. Las características típicas de los vehículos son:

1. Cada vehículo parte de un depósito, pero puede terminar su ruta en un depósito diferente.
2. Los vehículos tienen una capacidad definida.

3. Los vehículos pueden subdividirse y caracterizarse con base en el tipo de demanda que llevan.

4. Hay un costo asociado a la utilización de los vehículos.

Las características típicas de los clientes son:

1. Los vértices de la gráfica en donde se encuentran los clientes.

2. Cantidad solicitada y tipo (en caso que hubiera), por cada cliente.

3. Periodos del día (ventanas de tiempo) durante los cuales el cliente puede recibir a los vehículos.

4. Subconjunto de vehículos que pueden satisfacer la demanda de cada cliente.

Los objetivos típicos son:

1. Minimizar el costo global de todos lo vehículos el cual depende de la distancia global de los mismos (o el tiempo global de viaje).

2. Minimizar el número de vehículos requeridos para cumplir las demandas de todos los clientes.

3. Balancear las rutas en términos del tiempo total de viaje de cada vehículo y los clientes a los que da servicio.

4. Minimizar las penalizaciones asociadas al realizar entregas parciales a los clientes.

5. La combinación convexa de algunos de los objetivos previamente mencionados.

2.1.2. Métodos Heurísticos

Existen un gran número de heurísticas para resolver el VRP, sin embargo, se pueden clasificar en dos variantes las cuales son: heurísticas clásicas y metaheurísticas.

Las heurísticas clásicas pueden clasificarse en tres categorias. La primera son heurísticas constructivas, las cuales como su nombre lo indica van construyendo poco a poco una solución factible tomando en cuenta el costo de ésta, sin embargo, no contiene ningún procedimiento adicional para mejorar la calidad de la solución.

La segunda categoría corresponde a las heurísticas de dos fases, en la primera fase se busca crear conglomerados de clientes con los cuales pueda crear una ruta viable, y despúes se crea propiamente la ruta que seguirán los vehículos. Las heurísticas de dos fases se dividen en dos clases: la primera corresponde a las heurísticas en las que se crea primero el conglomerado de clientes y después se crea la ruta, y la segunda a las heurísticas donde se crea primero la

ruta y después se hacen los conglomerados. En el primer caso los clientes son organizados en posibles conglomerados y después se crea una ruta para cada uno de ellos, en el segundo caso se crea una ruta que incluye a todos los clientes y después se va seccionando hasta obtener rutas viables.

La tercer categoria corresponde a las heurísticas mejoradas, que modifican cualquier solución factible mediante el intercambio de clientes ya sea dentro de la misma ruta o en rutas diferentes.

En cuanto a las metaheurísticas, éstas se enfocan en hacer una búsqueda exhaustiva en las regiones más prometedoras del espacio de solución. La calidad de las soluciones que se obtienen mediante este método se espera sean de mejor calidad que las obtenidas mediante heurísticas clásicas.

2.1.3. Métodos Exactos

Como sabemos el VRP es *NP-duro* por lo que, que algún método exacto pueda resolver el problema en un tiempo de cómputo razonablemente bueno dependerá de qué tan complejo sea el problema que se intenta resolver. Los métodos exactos para resolver el VRP pueden clasificarse en dos diferentes:

1. Programación Dinámica.
2. Programación lineal entera.

Mucho del trabajo en programación lineal entera se basa en las formulaciones descritas en la sección 2.1.1. El método de ramificación y corte ha sido utilizado en las últimas décadas para resolver la variante del problema de ruteo vehícular con capacidades. Laporte y Norbert [9] hacen un análisis completo de los algoritmos de ramificación y corte propuestos a finales de 1980. A partir de éstos, los métodos exactos más efectivos para resolver el problema de ruteo vehícular con capacidades fueron los algoritmos de ramificación y corte utilizando como base algunas relajaciones del problema de asignación, los últimos algoritmos propuestos para resolver esta variante del problema obtenían resultados exactos cuando se tenían matrices de costo tanto simétricas como asimétricas.

2.2. Problema de Ruteo del Autobús Escolar

La problematica de cómo transportar a los estudiantes a las escuelas de la forma más segura, económica, equitativa y conveniente, trajo como consecuencia que se planteara el problema de ruteo del autobús escolar (SBRP). Al momento de resolverse uno de los objetivos que se plantea es encontrar las rutas hacia la escuela que lleven a los estudiantes de forma económica, segura y equitativa, ya sea en un entorno rural o urbano. Por ejemplo en

un entorno urbano, se tiene una densidad de población más alta, la distancia que existe en cada ruta es menor, la cantidad de estudiantes en los puntos de recolección aumenta y los autobuses pueden ser guardados en un depósito. En un entorno rural, hay menos densidad de población, rutas de autobús más largas, así como menos estudiantes en los puntos de recolección, por lo que se requiere de más paradas por ruta.

Generalmente el SBRP consiste en los siguientes subproblemas:

1. Seleccionar el número de autobuses a utilizar.
2. Seleccionar las paradas de autobús, a partir de un posible conjunto de éstas.
3. Asignar los estudiantes a las paradas de autobús.
4. Asignar las paradas (de autobús) a los autobuses.
5. Determinar las rutas de los autobuses.

El SBRP es más complicado que el VRP, sin embargo, dada la cantidad y ubicación de los puntos de recolección, más la cantidad de estudiantes asignados a cada uno de ellos, el SBRP se reduce a un problema de ruteo vehícular como el presentado en la sección 2.1, por lo tanto el SBRP es un problema *NP-duro*.

Cada versión del problema que se cita en la literatura tiene sus propias peculiaridades y se hacen diferentes suposiciones, objetivos y restricciones, por ejemplo, sólo unos pocos artículos consideran todos los subproblemas previamente mencionados, por lo que no existe ningún método que esté por encima de otro para resolver cada versión del problema, incluso algunos de los métodos son específicos para una versión particular por lo que no tiene un método que pueda resolver cada versión existente.

2.2.1. Descripción del problema y formulaciones

El SBRP consiste en encontrar un conjuto de rutas óptimas, las cuales serán recorridas por los autobuses, tal que cumplen las siguientes restricciones:

1. Cada autobús recorre una única ruta.
2. Cada ruta empieza en la escuela y termina en la escuela.
3. Cada parada de autobús (punto de recolección) es visitado por un sólo autobús.
4. El número de estudiantes en cada autobús no puede superar la capacidad del mismo.
5. El tiempo de viaje (o la distancia que recorre) no puede superar lo máximo permitido.

Las siguientes formulaciones que se revisan resuelven sólamente algunos de los subproblemas antes mencionados, así como sólo utilizan algunas de las suposiciones iniciales.

Bowerman, Hall y Calamai [4] proponen una formulación multiobjetivo para el SBRP considerando un entorno urbano, buscando resolver los subproblemas 2 al 5 y respetando las restricciones de la 2.1 a la 2.4.

Formulación de Bowerman, Hall y Calamai

$$min \qquad \{f_1, f_2, f_3, f_4\} \tag{2.1}$$

$$s.a. \quad c_{ij}z_{ij} \leq S_j, \ \forall i \in B; j \in J \tag{2.2}$$

$$z_{ij} \leq u_i, \ \forall i \in B; j \in J \tag{2.3}$$

$$\sum_{i \in B} z_{ij} = 1, \ \forall j \in J \tag{2.4}$$

$$\sum_{i \in B} (y_{ik} \sum_{j \in J} v_j z_{ij}) \leq W, \ \forall k \in K \tag{2.5}$$

$$\sum_{k \in K} y_{ik} = \begin{cases} |K|, & i \in S; \\ u_i, & i \in B \end{cases} \tag{2.6}$$

$$\sum_{i \in I} x_{ijk} = y_{jk}, \ \forall j \in I; k \in K \tag{2.7}$$

$$\sum_{j \in I} x_{ijk} = y_{ik}, \ \forall i \in I; k \in K \tag{2.8}$$

$$\sum_{l \in B} f_{ilk} - \sum_{l \in B} f_{lik} = \sum_{j \in B} x_{ijk}, \ \forall i \in B; k \in K \tag{2.9}$$

$$f_{ijk} \leq n x_{ijk}, \ \forall j \in I; k \in K \tag{2.10}$$

$$f_{ijk} \geq 0, \ \forall i, j \in I; k \in K \tag{2.11}$$

$$u_i \in \{0, 1\}, \ \forall i \in B \tag{2.12}$$

$$x_{ijk} \in \{0, 1\}, \ \forall i, j \in I; k \in K \tag{2.13}$$

$$y_{ik} \in \{0, 1\}, \ \forall i \in I; k \in K \tag{2.14}$$

$$z_{ij} \in \{0, 1\}, \ \forall i \in B; j \in J \tag{2.15}$$

Donde S es un conjunto de cardinalidad uno que representa a la escuela, B es el conjunto de todas las posibles paradas de autobús, $I = S \cup B$ representa todos los posibles puntos existentes, J es el conjunto de todos los estudiantes, K es el conjunto de todos los autobuses, W es la capacidad que tienen los autobuses, c_{ij} es la distancia que el estudiante $j \in J$ camina hasta la ubicación $i \in I$ o la distancia que hay entre dos ubicaciones $i \in I$ y $j \in I$, S_j es la

distancia máxima que puede caminar el estudiante $j \in J$ hasta una parada de autobús, v_j es el peso asignado al estudiante $j \in J$, $n = |I|$ es la cantidad de puntos a rutear, la variable binaria $u_i = 1$ si una parada de autobús es asignada a la ubicación $i \in B$, $u_i = 0$ en otro caso, la variable binaria $x_{ijk} = 1$ si el punto $i \in I$ precede inmediatamente al punto $j \in I$ en la ruta $k \in K$, $x_{ijk} = 0$ en otro caso, la variable binaria $y_{ik} = 1$ si el punto $i \in I$ es atendido por el vehículo $k \in K$, $y_{ik} = 0$ en otro caso y la variable binaria $z_{ij} = 1$ si el estudiante $j \in j$ es asignado a la parada de autobús $i \in B$, $z_{ij} = 0$ en otro caso. La función objetivo (2.1) esta compuesta por la distancia total de las rutas f_1, la distancia que caminan los estudiantes f_2, el balance en la carga de los autobuses f_3 y el balance en la distancia que recorren f_4, donde:

$$f_1 = \sum_{i,j \in I; k \in K} c_{ij} x_{ijk} \tag{2.16}$$

$$f_2 = \sum_{i \in B; j \in J} c_{ij} z_{ij} \tag{2.17}$$

$$f_3 = \sum_{k \in K} \left(\sum_{i \in B; j \in J} v_j y_{ik} z_{ij} \right)^2 \tag{2.18}$$

$$f_4 = \sum_{k \in K} \left(\sum_{i,j \in i} c_{ij} x_{ijk} - \frac{\sum_{ijk} c_{ij} x_{ijk}}{|K|} \right)^2 \tag{2.19}$$

El conjunto de restricciones (2.2) aseguran que los estudiantes no superen la distancia máxima que pueden caminar, el grupo de restricciones (2.3) aseguran que los estudiantes sean enviados a las paradas de autobús que en realidad se van a utilizar, los grupos de restricciones (2.4) garantizan que todo estudiante sea asignado a una parada de autobús, los grupos de restricciones (2.5) son las restricciones de capacidad del autobús, el grupo de restricciones (2.6) aseguran que cada parada de autobús que tenga al menos un estudiante sea asignada a una ruta y que la escuela está en todas las rutas, el grupo de restricciones (2.7) y (2.8) garantizan que para cada parada de autobús a la que se llega también se sale de ella y que sólo visita las paradas de autobús que están en su ruta, el grupo de restricciones (2.9), (2.10) y (2.11) previenen que se formen subciclos mediante el uso de la variables de flujo (f_{ijk}) y el grupo de restricciones (2.12) - (2.15) son las restricciones de integralidad de las varibles.

Li y Fu [11] también dan una formulación multiobjetivo para el problema, en la cual asumen que se conocen el número y la ubicación de las paradas de autobús, cada parada de autobús tiene asignada una cantidad de estudiantes y utiliza autobuses heterogéneos. La formulación dada por Li y Fu resuelve los subproblemas 4 y 5, satisfaciendo los grupos de restricciones 2, 3 y 4 usando K autobuses heterogéneos.

Formulación de Li y Fu

$$min\{ \sum_{k=1}^{K} (\sum_{i=1}^{n} (\sum_{j=1}^{n+1} t_{ij} x_{ijk} (\sum_{l=1}^{i} z_{lk}) + L z_{ik} (\sum_{l=1}^{i} z_{lk}))), \sum_{k=1}^{K} (\sum_{i=1}^{n} (\sum_{j=1}^{n+1} t_{ij} x_{ijk} + L z_{ik})) \} \tag{2.20}$$

$$s.a. \quad \sum_{i=1}^{n} y_{ik} \leq C_k, \; \forall k = 1, \ldots, K \tag{2.21}$$

$$\sum_{k=1}^{K} y_{ik} \leq f_i, \; \forall i = 1, \ldots, n \tag{2.22}$$

$$\sum_{i=1}^{n} x_{ilk} - \sum_{j=1}^{n+1} x_{ljk} = 0, \; \forall l = 1, \ldots, n; k = 1, \ldots, K \tag{2.23}$$

$$\sum_{i=1}^{n} x_{i(n+1)k} = \sum_{j=1}^{n+1} x_{0jk} = 1, \; \forall k = 1, \ldots, K \tag{2.24}$$

$$\sum_{j=1}^{n+1} x_{ijk} \geq z_{ik}, \; \forall i = 1, \ldots, n; k = 1, \ldots, K \tag{2.25}$$

$$y_{ik} \leq f_i z_{ik}, \; \forall i = 1, \ldots, n; k = 1, \ldots, K \tag{2.26}$$

$$x_{ijk} \in \{0, 1\}, \; \forall j = 1, \ldots, n; k = 1, \ldots, K \tag{2.27}$$

$$z_{ik} \in \{0, 1\}, \; \forall i = 1, \ldots, n; k = 1, \ldots, K \tag{2.28}$$

$$y_{ik} \geq 0, \; \forall i = 1, \ldots, n; k = 1, \ldots, K \tag{2.29}$$

Donde K es el número de autobuses disponibles, C_k es la capacidad del autobús k, n es el número de paradas de autobuses, M es el número de estudiantes, t_{ij} es el tiempo de viaje desde la parada de autobús i hasta la parada de autobús j, f_i es la cantidad de estudiantes que hay en la parada de autobús i, L es el tiempo promedio que se tarda en cada parada de autobús, 0 y $n+1$ representan a la escuela. La variable binaria $x_{ijk} = 1$ si el autobús k viaja de la parada de autobús i hasta la parada de autobús j, $x_{ijk} = 0$ en otro caso, la variable binaria $z_{ik} = 1$ si el autobús k recoge a los estudiantes de la parada de autobús i, $z_{ik} = 0$ en otro caso, y la variable y_{ik} denota la cantidad de estudiantes que recoge el autobús k en la parada de autobús i. La función objetivo (2.20) es minimizar el tiempo total de viaje de todos los estudiantes hasta los puntos de recolección y el tiempo total de viaje de los autobuses, el grupo de restricciones (2.21) son de capacidad del autobús, el grupo de restricciones (2.22) aseguran que los estudiantes en la parada de autobús i deben ser recogidos por al menos un autobús, el grupo de restricciones (2.23) garantizan que todo autobús que llega a una parada de autobús también sale de ella, el grupo de restricciones (2.24) garantizan que todos los autobuses visitan a la escuela, el grupo de restricciones (2.25) aseguran que si un autobús recoge estudiantes de un punto de recolección, debe de visitar ese punto de recolección, el grupo de restricciones (2.26) aseguran que el autobús k sólo recoge a los estudiantes de la parada de autobús i que han sido asignados al autobús k.

2.2.2. Métodos Heurísticos

Newton y Thomas [12] propusieron un método heurístico, mediante la creación de una solución inicial del problema del agente viajero particionado en rutas, donde cada ruta es recorrida por un autobús, en donde se busca minimizar la cantidad de rutas requeridas y la distancia total de viaje de los autobuses. Tiempo después los mismos Newton y Thomas [13] consideraron el ruteo de los autoubuses donde se atiende a más de una escuela.

Bowerman, Hall y Calamai [4] proponen un método heurístico con tres procedimientos. El primer procedimiento es encontrar P cubiertas de conjuntos de las paradas de autobús para cada ruta, de tal manera que cada estudiante es asignado a una parada de autobús dentro de su radio de distancia en la cual puede caminar. El segundo procedimiento es encontrar un subtour para cada conjunto de rutas que se obtuvieron en el primer procedimiento. El tercer procedimiento consiste en aplicar la heurísta de inserción de paradas de autobús a la solución obtenida en el segundo procedimiento.

Braca, Bramel, Posner y Simchi-Levi [5] implementaron una versión aleatoria de la heurística basada en ubicación (LBH[3]), fue propuestó originalmente para resolver el problema de ruteo vehicular con capacidades. Li y Fu [11] propusieron un método heurístico con cinco procedimientos. El propósito del primer procedimiento es encontrar el óptimo del problema:

$$K = min\, q \tag{2.30}$$

$$s.a. \quad \sum_{k=1}^{q} C_k \geq M \tag{2.31}$$

El cuál es minimizar la cantidad de autobuses que se requieren para visitar todos los puntos. El propósito del segundo procedimiento es crear una solución inical para el problema. El tercer procedimiento busca mejorar la solución inicial asignando estudiantes que están en los autobuses que recorren una distancia grande a autobuses que recorren una distancia corta. El cuarto procedimiento consiste en ajustar la solución creada en el paso anterior. El quinto procedimiento consisite en resolver apropiadamente el problema de asignación el cual minimiza el tiempo que el autobús tiene asientos vacantes para los autobuses que tienen la misma capacidad mediante el uso del método Húngaro.

2.2.3. Métodos Exactos

Ya que el SBRP es un problema *NP-duro*, la mayoría de los trabajos existentes se enfocan en encontrar una solución utilizando métodos heurísticos, por lo que los trabajos que dan una solución mediante un método exacto son prácticamente inexistentes, sin embargo, Bektas y Elmastas [2] resuelven una versión del SBRP mediante el uso de programación lineal entera.

[3]Location Based Heuristic

$$min \quad \sum_{i \in V'} \sum_{j \in V'} c_{ij} x_{ij} + f k \tag{2.32}$$

$$s.a. \quad \sum_{i \in I} x_{0i} \leq k \tag{2.33}$$

$$\sum_{i \in I} x_{id} \leq k \tag{2.34}$$

$$\sum_{j \in I \cup \{d\}} x_{ij} = 0, \; \forall i \in I \tag{2.35}$$

$$\sum_{i \in I \cup \{0\}} x_{ij} = 1, \; \forall j \in I \tag{2.36}$$

$$u_i - u_j + Q x_{ij} + (Q - q_i - q_j) x_{ij} \leq Q - q_j, \; \forall i \neq j \in I \tag{2.37}$$

$$u_i \geq q_i, \; \forall i \in I \tag{2.38}$$

$$u_i - q_i x_{0i} + Q x_{0i} \leq Q, \; \forall i \in I \tag{2.39}$$

$$v_i - v_j + (T - d_{id} - d_{0j} + d_{ij}) x_{ij} + (T - d_{id} - d_{0j} - d_{ji}) x_{ji} \leq T - d_{id} - d_{0j} \quad \forall i \neq j \in I \tag{2.40}$$

$$v_i - d_{0i} x_{0i} \geq 0, \; \forall i \in I \tag{2.41}$$

$$v_i - d_{0i} x_{0i} + T x_{0i} \leq T, \; \forall i \in I \tag{2.42}$$

$$x_{ij} \in \{0, 1\}, \; \forall i, j \in V' \tag{2.43}$$

El problema se formula en una gráfica completa $G = (V, A)$ con $|V|$ nodos y $|A|$ arcos, la distancia entre cada par de nodos está dada por una matriz simétrica $D := [d_{ij}]$ donde d_{ij} representa la distancia que hay del nodo i hasta el nodo j. El conjunto de nodos está particionado $V = \{0\} \cup I$, donde el nodo 0 es el depósito e I es el conjunto de nodos intermedios, cada nodo intermedio $i \in I$ tiene una cantidad de estudiantes (denotado por q_i) que deben ser recogidos. Q es la capacidad que tienen los autobuses y T es la distancia máxima que pueden tener las trayectorias, k es la cantidad de trayectorias disjuntas que se quiere minimizar. Para el problema se introduce un nuevo nodo d el cual se conecta a los nodos terminales de las trayectorias, de tal manera que el problema se reduce en encontrar k trayectorias disjuntas entre dos puntos en la gráfica extendida $G' = (V', A')$, donde $V' = V \cup d$ y las nuevas distancias de A' se definen como:

$$d'_{ij} = \begin{cases} 0 & \text{si } i \in I \text{ y } j = d \\ M & \text{si } i \in 0 \text{ y } j = d \\ d_{ij} & \text{en otro caso} \end{cases}$$

c_{ij} es el costo de pasar por el arco (i, j) y es típicamente una función de distancia $c_{ij} = \alpha d_{ij}$, donde α es el costo por unidad de distancia, f es la unidad de costo por cada autobús, la variable u_i representa la cantidad de estudiantes que el autobús recoge después de pasar por el nodo i, la variable v_i denota la distancia que el autobús ha recorrido hasta el punto i, $x_{ij} = 1$ si el arco (i, j) se utiliza en la solución.

La función objetivo (2.32) busca minimizar el costo total de viaje y el costo del número de autobuses utilizados, el grupo de restricciones (2.33) y (2.34) aseguran que a lo más k vehículos partan del nodo 0 y lleguen al nodo d, el grupo de restricciones (2.35) y (2.36) aseguran que cada nodo intermedio sea visitado una sola vez, el grupo de restricciones (2.37), (2.38) y (2.39) son de capacidad y el grupo de restricciones (2.40), (2.41) y (2.42) son de distancia.

En 2013 Kim y Park [10] construyen una formulación, que fácilmente se puede transformar en un programa lineal entero mixto, para resolver una versión del problema con ventanas de tiempo considerando una sóla escuela, además se supone que se conoce la ubicación de los puntos de recolección así como la cantidad de estudiantes que hay en cada punto y donde todos los autobuses utilizados tienen la misma capacidad.

Las variables de decisión asociadas al problema son:

$$x_{ij}^k = \begin{cases} 1 & \text{si el vehículo } k \text{ pasa por el arco ij} \\ 0 & \text{en otro caso} \quad k \in V, i \in DS, j \in SE. \end{cases}$$

$t_i^k =$ el tiempo actual en el cual el vehículo k llega al nodo $i, i \in TD, k \in V$
Las constantes son:

$F =$ costo que tiene cada autobús.
$R =$ costo de la ruta por unidad de tiempo.
$t_{ij} =$ tiempo que se tarda de ir desde el nodo i hasta el nodo j, $i \in DS, j \in SE$.
$p_i =$ demanda (estudiantes) en el nodo i, $i \in S$.
$Q_k =$ capacidad del vehículo k, $k \in V$.
$b =$ tiempo promedio que tarda un estudiante en subir al autobús.
$T_e =$ menor tiempo en el cual el autobús puede llegar a la escuela.
$T_l =$ tiempo máximo en el cual el autobús puede llegar a la escuela.
$T =$ tiempo máximo permitido a los vehículos.

Los conjuntos de datos son:

$S =$ nodos con demanda (paradas de autobús)
$D =$ nodo donde empiezan los vehículos (puede ser un depósito).
$E =$ nodo donde llegan todos los vehículos (la escuela).

DS = todos los nodos que permiten salir de ellos (el depósito y las paradas de autobús) $D \cup S$.
SE = todos los nodos a los que se puede entrar (la escuela y las paradas de autobús) $E \cup S$.
TD = el conjunto de todos los nodos, $D \cup S \cup E$.
V = el conjunto vehículos a utilizar.

$$min \quad Z = F \sum_{k \in V} \sum_{i \in D} \sum_{j \in S} x_{ij}^k + R \sum_{i \in DS} \sum_{j \in SE} \sum_{k \in V} x_{ij}^k t_{ij} \tag{2.44}$$

$$s.a. \quad \sum_{k \in V} \sum_{i \in DS} x_{ij}^k = 1,\ j \in S \tag{2.45}$$

$$\sum_{k \in V} \sum_{j \in SE} x_{ij}^k = 1,\ i \in S \tag{2.46}$$

$$\sum_{i \in DS} x_{ip}^k - \sum_{j \in SE} x_{pj}^k = 0,\ k \in V, p \in S \tag{2.47}$$

$$\sum_{i \in D} \sum_{j \in S} \sum_{k \in V} x_{ij}^k - \sum_{i \in S} \sum_{j \in E} \sum_{k \in V} x_{ij}^k = 0 \tag{2.48}$$

$$\sum_{i \in DS} p_i \Big(\sum_{j \in SE} x_{ij}^k \Big) \leq Q_k,\ k \in V \tag{2.49}$$

$$t_i^k + t_{ij} + p_i \times b \leq t_j^k + M(1 - x_{ij}^k),\ k \in V, i \in DS, j \in SE \tag{2.50}$$

$$t_i^k + t_{ij} + p_i \times b \geq t_j^k + M(x_{ij}^k - 1),\ k \in V, i \in DS, j \in SE \tag{2.51}$$

$$T_e \leq t_E^k \leq T_l \tag{2.52}$$

$$t_i^k + p_i \times b \geq t_E^k - T,\ i \in S, k \in V \tag{2.53}$$

$$\sum_{k \in V} \sum_{j \in S} x_{ij}^k \leq 1,\ i \in D \tag{2.54}$$

$$\sum_{k \in V} \sum_{j \in S} x_{ij}^k \leq 1,\ j \in E \tag{2.55}$$

Donde la función objetivo (2.44) busca minimizar el costo total de recoger a todos los estudiantes. El grupo de restricciones (2.45) y (2.46) aseguran que cada nodo sea atendido por un solo vehículo, el grupo de restricciones (2.47) aseguran que a cada nodo que se entra se sale de él, el grupo de restricciones (2.48) aseguran que todos los vehículos que salen del depósito llegan a la escuela, el grupo de restricciones (2.49) aseguran que no se sobrepase la capacidad de los autobuses, el grupo de restricciones (2.50) - (2.52) aseguran que el autobús llegue a la escuela entre el tiempo T_e y el tiempo T_l, el grupo de restricciones (2.53) aseguran que ningún autobús salga de el nodo i antes de el tiempo de salida mínimo de ese nodo y el grupo de restricciones (2.54) y (2.55) aseguran que cada vehículo sale del depósito y llega

a la escuela una sola vez por lo que el número de rutas generadas debe ser menor o igual al número de autobuses utilizados.

Una vez obtenida la formulación, procedieron a probarla con algunas instancias y compararon sus resultados con los obtenidos mediante búsqueda armónica para las mismas instancias del problema, pero debido a la complejidad del modelo las instancias de prueba son pequeñas, como se muestra en la siguiente tabla:

# nodos	**# autobuses**	**Método exacto**			**Búsqueda armónica**		
		Tiempo(s)	**F.O.**	**# rutas**	**Tiempo(s)**	**F.O.**	**# rutas**
6(4)	2	0.16	210,500	2	0.15	210,500	2
6(4)	3	0.78	210,500	2	0.15	210,500	2
9(7)	3	1343	306,195	3	1.0	306,195	3
12(10)	4	7926	307,980	3	2.0	307,980	4
20(18)	6	-	-	-	129	517,745	5

Cuadro 2.1: Resultados computacionales del método exacto de Kim y Park

Capítulo 3

Formulación del problema de ruteo del autobús escolar con recolección mixta

3.1. Antecedentes

La versión del problema a resolver se puede considerar como un caso particular del problema del autobús escolar para estudiantes con necesidades especiales, el cual tiene aspectos puntuales que lo diferencian del problema para estudiantes regulares. Para empezar los estudiantes con necesidades especiales se recogen y se entregan (en caso de considerar esta opción dentro del planteamiento del problema) directamente en sus casas, existe una restricción más rígida en cuanto al tiempo (o distancia) que pasan los estudiantes con necesidades especiales arriba del autobús en comparación a los estudiantes regulares y finalmente cada estudiante debe tener una prioridad al momento de ser recogido dependiendo del tipo de discapacidad que éste tenga, ya que algunos estudiantes requieren de equipo o ayuda especial al momento de ser transportados, en ese caso se debe de tener un autobús especial para poder transportarlos, todo esto hace que muy pocos consideren estudiantes con necesidades especiales en el problema.

El problema fue introducido por primera vez por Russell y Morrel [15] en 1986, para tratar de dar una solución a las necesidades del sistema de transpote escolar en la ciudad de Tulsa en el estado de Oklahoma en EEUU. Al principio crean una solución inicial mediante una modificación a la heurística presentada por Clark y Wright [6] y mejoran esa solución mediante al algoritmo 3-OPT y la mejora del algoritmo M-TOUR.

Tiempo después en 1997 Braca *et al.* [5] discuten brevemente el rol que juegan los estudiantes con necesidades especiales dentro de un sistema de transporte escolar. Presentan el estatus del sistema de transporte escolar en la ciudad de Nueva York e incluso la diferencia que existe entre los estudiantes con necesidades especiales y los estudiantes regulares. Sin embargo se enfocan en resolver el problema solamente para los estudiantes regulares.

En 2005 Ripplinger [14] deja a un lado el entorno urbano y se enfoca en la problemática asociada a resolver el problema en un entorno rural y las consecuencias que conlleva el re-

coger a estudiantes con necesidades especiales a la par de los estudiantes regulares. Dada la estructura existente en un entorno rural donde la cantidad de estudiantes es baja y es usual recoger a los estudiantes casa por casa, asume que no es necesario resolver el problema de asignación de los estudiantes a los puntos donde serán recogidos. Propuso un modelo matemático para resolver el problema y desarrolló una nueva heurística para el mismo, la cual consta de dos partes: se construye una ruta inicial y posteriormente se mejora mediante el uso de una búsqueda tabú. Da dos opciones para resolver el problema, la primera consiste en distinguir entre los estudiantes con necesidades especiales y los estudiantes regulares, después los problemas son resueltos por separado. En la segunda opción genera una sola ruta la cual incluye a los dos tipos de estudiantes.

3.2. Descripción del problema

Además de las restricciones que se consideran para cada versión del problema, existen componentes adicionales las cuales dependen de quien éste a cargo de resolver el problema, por ejemplo, los directores de las escuelas estarían más interesados en minimizar la cantidad de autobuses que se emplean para dar el servicio, ya que cada uno de ellos genera un costo. Por otro lado la preocupación de los padres de familia radica en el tiempo que el estudiante está a bordo del autobús, por lo que si el problema depende de ellos buscarían minimizar este tiempo.

El punto de vista que se toma para la versión de nuestro problema se puede ver como un consenso entre los padres y las autoridades escolares, ya que desde un inicio se tiene la cantidad de autobuses a utilizar (que es la preocupación principal de las autoridades escolares) y se busca minimizar la distancia que recorrerán dichos autobuses (lo cuál es la preocupación de los padres de familia).

Tradicionalmente a la hora de resolver el problema se tienen dos supuestos con respecto a la forma de recoger a los estudiantes. El primero es que los estudiantes son recogidos directamente es sus casas, por lo regular este supuesto se utiliza en mayor medida cuando se habla de la versión rural del problema, dado que la cantidad de estudiantes que se tiene que recoger es poca y la distancia que existe entre ellos no permite utilizar paradas de autobús para poderlos reunir, como se muestra en la figura 3.1.

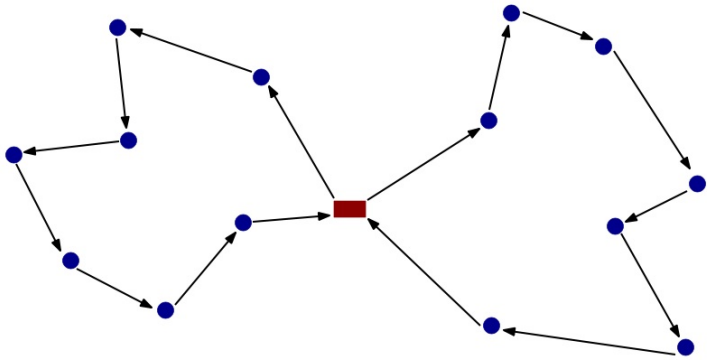

Figura 3.1: Ejemplo de estudiantes recogidos en sus casas.

En el segundo, los estudiantes son asignados a paradas de autobús para posteriormente ser recogidos por los autobuses, este supuesto se utiliza cuando se pretende resolver la versión urbana del problema, debido a que la cantidad de estudiantes a recoger es muy elevada y se vuelve impráctico pasar a recogerlos directamente en sus casas, pero ahora se vuelve viable el poder juntarlos en algún punto (comúnmente llamando parada de autobús) y de esta manera pasar a recogerlos, como se muestra en la figura 3.2.

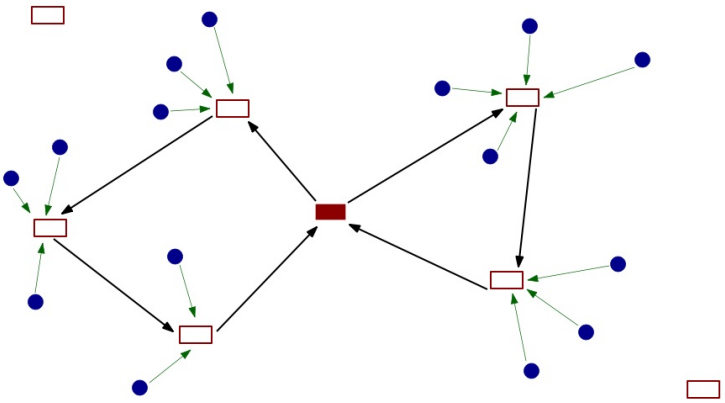

Figura 3.2: Ejemplo de estudiantes que caminan hasta un punto de recolección.

Para la versión del problema que se propuso resolver, si bien no se plantea propiamente la existencia de estudiantes con necesidades especiales, sí se especifica la existencia de dos tipos de estudiantes, el primer tipo son los estudiantes regulares los cuales pueden llegar caminando sin problemas a los puntos de recolección para ser recogidos posteriormente por los autobuses, el segundo tipo son los estudiantes especiales los cuales por algún motivo (ya sea alguna discapacidad, algún factor social o por motivos de seguridad) no puede llegar caminando a algún punto de recolección y tiene que ser recogido directamente en su casa.

Debido a que se están considerando a estos dos tipos de estudiantes, aún es necesario realizar la asignación de los estudiantes del primer tipo a los puntos de recolección para que puedan ser recogidos. A diferencia de la mayoría de los problemas de ruteo existentes en la literatura donde lo que se busca es que las rutas generadas sean ciclos, lo que nosotros buscamos es que las rutas sean trayectorias.

La función objetivo propuesta busca minimizar la distancia total que recorren todos los autobuses, así como la distancia que recorren los estudiantes a los puntos de recolección (los que pueden hacerlo). Debido que la asignación de los estudiantes influye directamente en los puntos que se tienen que visitar y esto a su vez influye en la distancia total que recorrerán los autobuses, si se realiza la asignación de estudiantes sin ninguna restricción esto provocaría que los estudiantes pudieran ser asignados a paradas de autobuses que físicamente sean imposibles de alcanzar por lo que para evitar que esto ocurra cada estudiante solo tiene las paradas de autobuses que son realmente alcanzables por ellos y se ignoran las demás.

De esta forma se reduce la cantidad de combinaciones posibles que se pueden hacer al momento de hacer la asignación de los estudiantes y deja a éstos con solo algunas opciones, en el caso que el estudiante sólo tenga una opción posible a donde caminar, entonces ese punto de recolección se vuelve obligatorio visitarlo y los estudiantes que puedan llegar a ese punto (obligatorio) se asignan para que lleguen a él, incluso si no es su mejor opción disponible, como se muestra en la figura 3.3.

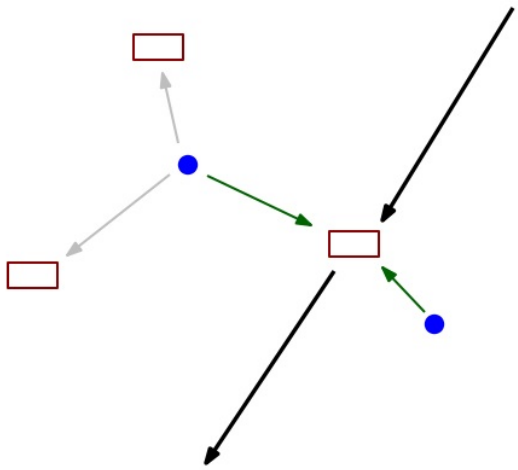

Figura 3.3: Ejemplo de asignación cuando un estudiante sólo tiene un punto de recolección al cual ir y los demás estudiantes se asignan a ese punto de recolección. Las flechas verdes denotan a qué punto fue asignado, mientras que las flechas grises indican las demás posibilidades de asignación y las flechas negras la ruta que sigue el autobús.

Tambien existe la posibilidad que los demás estudiantes no se asignen a ese punto de recolección (obligatorio) y sean asignados a otro que les convenga, pero en este caso debido a que la ruta que sigue el autobús visita alguno de los puntos de recolección que tienen disponibles los demás estudiantes de tal manera que la distancia recorrida por el autobús no se incrementa por asignar a los estudiantes a un punto de recolección diferente, como se muestra en la figura 3.4.

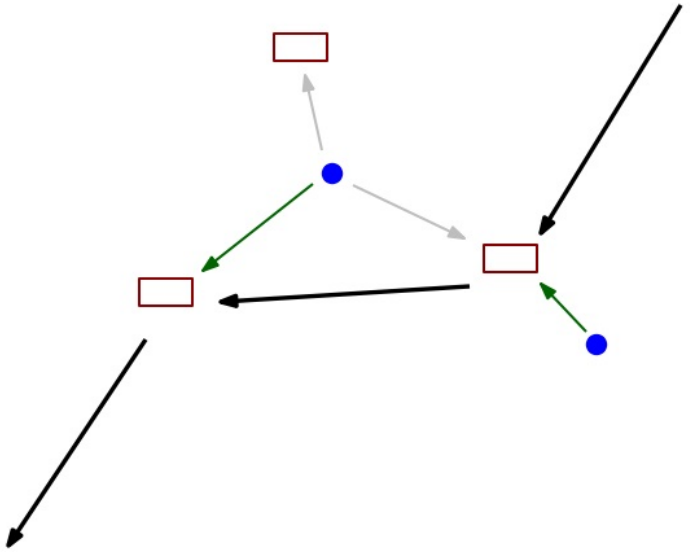

Figura 3.4: Ejemplo de asignación cuando un estudiante solo tiene un punto de recolección al cual ir y los demás estudiantes se asignan a un punto de recolección diferente. Las flechas verdes denotan a qué punto fue asignado, mientras que las flechas grises indican las demás posibilidades de asignación y las flechas negras la ruta que sigue el autobús.

Se asume que se conoce la cantidad de autobuses que se tiene disponible, así como la capacidad de los mismos, la distancia que hay entre la ubicación inicial de los estudiantes a los posibles puntos de recolección y la distancia que hay entre todos los posibles puntos a visitar (incluyendo al depósito y a la escuela).

Las restricciones para el problema son: todo autobús empieza su recorrido en el depósito y termina en la escuela, la cantidad de estudiantes que son recogidos por un autobús no puede superar la capacidad del mismo, cada punto de recolección es visitado por a lo más un autobús, cada ruta es visitada por un solo autobús, cada ruta contiene al menos una punto de recolección a visitar (el cual es diferente de la escuela y el depósito) y la distancia que los estudiantes caminan hasta un punto de recolección no puede exceder el valor máximo de la distancia que pueden caminar.

3.3. Método exacto

Para obtener una solución exacta del problema, se decidió modelar el problema como de programación lineal entera. Si bien no se encontró literatura relacionada con el problema en su totalidad de forma exacta, es decir, resolver la parte de asignación de los estudiantes así como la generación de rutas, sí existen formulaciones matemáticas que se usan como base para crear una heurística que resuelva el problema. Se decidió estudiar algunas de estas formulaciones para determinar si alguna de ellas pudiera ser de utilidad para adaptarse al problema que se planteó resolver.

Además de las restricciones mencionadas anteriormente se agregó una restricción adicional para evitar que se formen subciclos a lo largo de las trayectorias.

Finalmente se optó por realizar un preprocesamiento de los datos, que como ya se mencionó anteriormente dicho preprocesamiento consiste en reducir la cantidad de puntos de recolección asociados a cada estudiante de tal manera, que sólo queden relacionados con él dichos puntos de recolección que puede alcanzar, de esta manera se reducen sustancialmente las combinaciones que se pueden formar al momento de asignar a los estudiantes y con esto se asegura que ningún estudiante sea asignado a un punto de recolección al cual le sea físicamente imposible llegar.

Los elementos necesarios para poder llevar a cabo la formulación son los siguientes:

Conjuntos

S, $|S| = 2$, Donde S_0 Depósito, S_1 Escuela.
B, Conjunto de posibles puntos de recolección.
B_i, Conjunto de posibles puntos de recolección más cercanos a la ubicación del estudiante i, $i \in I$.
I, Conjunto de estudiantes a recoger. $I = I' \cup I''$
I', Conjunto de estudiantes que caminan a un punto de recolección.
I'', Conjunto de estudiantes que son recogidos en su casa.
M, Conjunto de autobuses a utilizar.
D, Conjunto de todos los posibles puntos a recorrer, $D = B \cup I'' \cup S$.
D', Conjunto de todos los puntos internos a recorrer $D' = B \cup I''$

Parámetros

W, Capacidad de los autobuses.
d_{ij}, Distancia que hay desde la casa del estudiante 'i' hasta el punto de recolección j. $i \in I, j \in B$

c_{ij}, Distancia que hay desde el punto $i \in D$ hasta el punto $j \in D$.

Variables

y_{ij}, binaria, 1 si el estudiante 'i' camina hasta el punto de recolección 'j', 0 en otro caso.
x_{ijk}, binaria, 1 si el punto $i \in D$ precede inmediatamente al punto $j \in D$ en la ruta $k \in M$, 0 en otro caso.
z_{jk}, binaria, 1 si el punto $j \in D$ es atendido por el vehículo $k \in M$, 0 en otro caso.
u_j, binaria, 1 si el punto $j \in D'$ es atendido por algun autobús, 0 en otro caso.
e_{ijk}, binaria, 1 si el estudiante $i \in I$ es recogido en el punto $j \in D'$ por el autobús $k \in M$, 0 en otro caso.
f_{ijk}, entera, variable de flujo auxiliar para evitar subrecorridos.

$$Min\{\sum_{i\in I'}\sum_{j\in B_i} d_{ij}y_{ij} + \sum_{i\in D}\sum_{j\in D}\sum_{k\in M} c_{ij}x_{ijk}\} \tag{3.1}$$

$$s.a. \quad \sum_{j\in B_i} y_{ij} = 1 \quad \forall i \in I' \tag{3.2}$$

$$y_{ii} = 1 \quad \forall i \in I'' \tag{3.3}$$

$$y_{ij} \leq u_j \quad i \in I, j \in D' \tag{3.4}$$

$$\sum_{j\in D'}\sum_{i\in I} e_{ijk} \leq W \quad k \in M \tag{3.5}$$

$$y_{ij} = \sum_{k\in M} e_{ijk} \quad i \in I, j \in D' \tag{3.6}$$

$$z_{jk} \geq e_{ijk} \quad i \in I, j \in D', k \in M \tag{3.7}$$

$$\sum_{i\in D} x_{ijk} = z_{jk} \quad j \in D, k \in M \tag{3.8}$$

$$\sum_{j\in D} x_{ijk} = z_{ik} \quad i \in D, k \in M \tag{3.9}$$

$$\sum_{k\in M} z_{jk} = |M| \quad j \in S \tag{3.10}$$

$$\sum_{k \in M} z_{jk} = u_j \quad j \in D' \tag{3.11}$$

$$\sum_{l \in D'} f_{ilk} - \sum_{l \in D'} f_{lik} = \sum_{j \in D'} x_{ijk} \quad i \in D', k \in M \tag{3.12}$$

$$f_{ijk} \leq |D'| x_{ijk} \quad i, j \in D, k \in M \tag{3.13}$$

$$f_{ijk} \geq 0 \quad i, j \in D, k \in M \tag{3.14}$$

$$y_{ij} = \{0, 1\} \quad \forall i \in I, j \in B \tag{3.15}$$

$$u_j = \{0, 1\} \quad j \in D' \tag{3.16}$$

$$x_{ijk} = \{0, 1\} \quad \forall i, j \in D, k \in M \tag{3.17}$$

$$z_{jk} = \{0, 1\} \quad \forall i \in D, k \in M \tag{3.18}$$

$$e_{ijk} = \{0, 1\} \quad \forall i \in I, j \in D', k \in M \tag{3.19}$$

La ecuación (3.1) es la función objetivo la cual está minimizando la distancia recorrida por los estudiantes a los puntos de recolección así como la distancia que recorren los autobuses. El conjunto de ecuaciones (3.2) garantizan que los estudiantes que caminan hasta un punto de recolección son asignados a un solo punto de recolección. El conjunto de ecuaciones (3.3) garantizan que los demás estudiantes (los cuales no fueron asignados a un punto de recolección) serán recogidos directamente en sus casas.

El conjunto de ecuaciones (3.4) garantizan que los puntos de recolección con al menos un estudiante sean visitados por algun autobús. El conjunto de ecuciones (3.5) garantizan que no se exceda la capacidad de los autobuses. El conjunto de ecuaciones (3.6) y (3.7) indican qué autobús va a recoger a cada estudiante.

El conjunto de ecuaciones (3.8) y (3.9) garantizan que en cada punto de recolección (donde haya estudiantes) un autobús va a llegar y salir de éste. El conjunto de ecuaciones (3.10) y (3.11) aseguran que cada autobús tiene asignada una única ruta. El conjunto de ecuaciones (3.12) es una restricción de flujo la cual evita que se formen subciclos a lo largo de cada ruta. El conjunto de ecuaciones (3.13) a (3.19) son restricciones de integralidad de las variables.

La gran cantidad de variables y restricciones que se agregaron para lograr que la formulación fuese lineal y así poder resolverla mediante un programa lineal, limitan un poco el tamaño

de las instancias que se pueden resolver de forma exacta, sin embargo tales restricciones abren la posibilidad de utilizar la misma formulación para versiones menos rígidas del problema, hablaremos un poco más de estas versiones en el capítulo 4.

Lo anterior trae como consecuencia que se busquen métodos alternativos para encontrar soluciones a intancias del problema en un tiempo razonable a costa de sacrificar la optimalidad de la solución buscada.

3.4. Método heurístico

De la misma forma que en muchos artículos de la literatura, se parte de una formulación matemática para elaborar una heurística que pueda dar una solución para el problema en un tiempo razonable para instancias grandes del problema.

La mayoría de los autores crean una metaheurística con base en la formulación matemática que presentan, considerando una función multiobjetivo, en nuestro caso optamos por utilizar GRASP para obtener una solución del problema en vez de crear una nueva versión de una metaheurística, la razón principal es que nuestra función objetivo no se puede considerar multiobjetivo (si bien tiene dos argumentos en ella, dichos argumentos son hasta cierto punto compatibles lo que hace que la típica problemática de un problema multiobjetivo no se presente).

El algoritmo GRASP es un procedimiento de multi-arranque en donde cada paso consiste en una fase de construcción y una de mejora. En la fase de construcción se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial. Esta solución se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final.

En la fase de construcción se construye iterativamente una solución posible, considerando un elemento en cada paso. En cada iteración la elección del próximo elemento para ser añadido a la solución parcial viene determinada por una función glotona aleatorizada, es decir, esta función no selecciona el mejor candidado según la función glotona, sino que con el objeto de diversificar y no repetir soluciones en dos construcciones diferentes, se construye un lista con los mejores candidatos de entre los que se toma uno al azar.

En la fase de mejora se suele emplear un procedimiento de intercambio simple con el objeto de no emplear mucho tiempo en esta mejora. Aqui se muestra el funcionamiento global del algoritmo:

Algoritmo 3.1 Algoritmo GRASP.

1: Mientras (Condición de parada)
2: **Fase Constructiva**
Seleccionar una lista de elementos candidatos.
Considerar una Lista Restringida de los mejores Candidatos.
Seleccionar un elemento aleatoriamente de la Lista Restringida.
3: **Fase de Mejora**
Realizar un proceso de búsqueda local a partir de la
solución construida hasta que no se pueda mejorar más.
4: **Actualización**
Si la solución obtenida mejora a la mejor almacenada, actualizarla.

Para la búsqueda local a utilizar en el GRASP se implementaron diversos tipos de búsqueda local para evaluar su rendimiento y utilizar la que mejor desempeño muestra, el número de iteraciones para la búsqueda local es de 500.

Las búsquedas locales que se probaron:

1. Intercambio de dos puntos dentro de una misma ruta.

2. Intercambio de dos puntos entre rutas distintas.

3. Cambio de un punto a una ruta distinta.

4. Intercambio del orden de un segmento en una misma ruta.

Antes de pasar a la construcción de la solución inicial para el GRASP se prueban las búsquedas locales mediante las cuales se pretende mejorar dicha solución inicial, para lo cual se probaron varias instancias para las búsquedas locales que previamente se mencionaron.

Instancia	**Rutas**	**Puntos**	**Búsqueda Local**				**Valor de la F.O.**
			1	2	3	4	
1	2	8	3296	3732	3811	3440	3296
2	2	12	5144	5417	6232	5517	5144
3	3	15	8635	9024	9238	8879	8635
4	3	21	9725	9982	10800	9915	9473
5	4	20	12365	13832	14137	12883	12082

Cuadro 3.1: Comparación entre las buśquedas locales implementadas

Si bien la búsqueda local que mostró un mejor desempeño fue el intercambio de dos puntos dentro de una misma ruta, el utilizar solamente esta búsqueda hace que sea difícil salir de óptimos locales, ya que no permite el intercambio de elementos entre rutas distintas, por lo que en la construcción inicial se deben de escoger correctamente qué elementos van en cada ruta, cosa que no se puede asegurar debido a que la solución inicial se realiza de forma estocástica.

En base a lo anterior en vez de tener una sola búsqueda local para mejorar la solución inicial se agrega una búsqueda local adicional, la cual es intercambiar dos elementos de rutas distintas, esta segunda búsqueda local solo se utiliza una vez que se llega a un óptimo local, es decir, con la combinación de elementos existente en cada ruta no es posible seguir mejorando la solución por lo que es necesario que de alguna manera se cambien los elementos existentes en las rutas de tal forma que se explore todo el espacio de búsqueda posible y se pueda encontrar una mejor solución.

Para construir la solución inicial para el GRASP se sigue considerando hacer un preprocesamiento de los datos de la misma manera que se realiza para el método exacto, recordemos que dicho preprocesamiento consiste en limitar los puntos de recolección a los cuales puede llegar un estudiante (los que pueden hacerlo) de tal manera que el estudiante ahora solo tiene un número limitados de paraderos a los cuales puede llegar, y dichos paraderos están delimitados por la proximidad existente con respecto al estudiante, esto se hace pensando que en un escenario real es ilógico que el estudiante se desplace grandes distancias solo para acceder a un paradero.

Una vez considerado esto la solución inicial consiste en asignar a los estudiantes(que puedan caminar) a un punto de recolección (no necesariamente el más cercano), una vez que se asignaron los estudiante ya se tienen los puntos a visitar (las casas de los estudiantes que no pueden llegar a un punto de recolección y los paraderos donde hay al menos un estudiante).

Una vez que tenemos identificados estos puntos, procedemos a formar las rutas, para ello utilizamos una lista restringida de cuatro elementos ya que con éste tamaño se obtuvieron los mejores resultados, ingresamos a ella los cuatro puntos más cercanos (con respecto al depósito), y se elige uno de forma aleatoria, de la misma forma para insertar este punto en alguna de las rutas, se elige de forma aleatoria a qué ruta se inserta, siempre y cuando que al insertar este punto no se viole la restricción de capacidad ya que la solución inicial tiene que ser una solución factible para el problema, y este proceso se repite hasta que el ultimo punto sea a signado a alguna ruta.

Algoritmo 3.2 Solución inicial del algoritmo GRASP. SolIni();

Entrada: Matrices de datos.

Salida: Una solución inicial donde estan asignados los estudiantes a los puntos de recolección y se tienen los puntos que se visitan en cada ruta.

1: Se elige la ubicación en donde serán recogidos los estudiantes.
Para los estudiantes que son recogidos en su casa se asigna la ubicación directamente **ubi[i] = 1**.
2: Para los estudiantes que caminan hasta un punto de recolección, se elige aleatoriamente el punto hasta donde camina **ubi[j] = 1** y se lleva el conteo de cuántos estudiantes fueron asignados hasta ese punto **cntubi[j]** ++.
3: Se crea una lista de candidados de tamaño cuatro en la cual se encuentran las ubicaciones más cercanas con respecto al depósito.
4: Se elige aleatoriamente un elemento de la lista de candidatos.
5: Se elige aleatoriamente la ruta en la cual se va a insertar, si al momento de insertarlo no se cumple con la restricción de capacidad, se elige una ruta distinta.
6: Se regresa al punto 3 hasta que se hayan insertado todas las ubicaciones a alguna ruta.

Para la búsqueda local mediante la cual vamos a mejorar la solución, una forma intuitiva de realizarla es que a cada paso de la iteración se seleccione una ruta y solo en esta ruta se realice el intercambio de elementos para mejorar el orden de visita de la ruta, sin embargo dependiendo de las cantidad de rutas que se estén considerando en la instancia el realizar la búsqueda local de esta forma puede resultar poco productivo y la cantidad de iteraciones necesarias para poder encontrar una solución de calidad puede ser demasiado alta incluso para instancias pequeñas del problema, por lo que en lugar de hacer el cambio en una sola ruta, el cambio se realiza en todas ellas a cada paso de la iteración, esto permite que a cada paso se tenga una configuración diferente de las rutas lo que nos lleva a tener una mejor exploración del espacio de búsqueda y la cantidad de iteraciones de la búsqueda local se ve reducido significativamente.

Para la segunda parte de la búsqueda local, como se mencionó anteriormente, una vez que se llega a un óptimo local, es decir, pasan un número grande de iteraciones sin que se mejore la solución, para nuestro caso consideramos este parámetro como 40 iteraciones sin cambio, es necesario el intercambio de elementos entre rutas distintas, pero en este caso sólo se realiza el intercambio entre dos rutas distintas ya que si bien puede parecer esto poco efectivo, si se hace un cambio drástico en la configuración de las rutas (cambian de elementos entre todas las rutas), es como si empezáramos nuevamente desde cero y se perdería el progreso que se lleva hasta el momento.

Si bien esta segunda búsqueda local nos permite explorar de mejor manera el espacio de solución, no nos garantiza que se llegue al óptimo, y la razón de ello radica en que no estamos permitiendo que se cambie el número de elementos que existen en cada ruta en ningún paso del algoritmo, por lo que si la cantidad de elementos que se obtiene en la solución no es el

correcto con respecto al óptimo, el ajuste de la segunda búsqueda local no es suficiente para llegar al óptimo, pero esto no implica que la solución entregada sea de mala calidad.

Para tratar de siempre llegar al óptimo se puede agregar una tercera búsqueda local en la cual se toma un punto de alguna ruta y se inserta en alguna otra, sin embargo, al momento de probar esta tercera búsqueda local en conjunto con las dos anteriores lo que resulta es en múltiples ocasiones al momento de trata de insertar un elemento en otra ruta esto no es posible ya que las rutas cumplen de forma exacta la restricción de capacidad por lo que es imposible insertar un elemento nuevo en una ruta sin antes sacar uno, otra situación que llega a ocurrir es el caso que las rutas se deforman de tal manera que existe una ruta que tenga muchos elementos que pesen poco y las demás poco elementos con un peso bastante considerable, por lo que aplicar esta tercera búsqueda local para intentar alcanzar el óptimo acarrea más problemas que beneficios.

Otro punto más para descartar esta tercera búsqueda local es que si bien las rutas no son homogéneas con respecto a la cantidad de puntos que visita cada una de ellas, sí se mantiene una estructura en la cual las rutas contienen más o menos la misma cantidad de elementos, por lo que recae en la configuración de las rutas que se generen en la solución inicial que se se pueda llegar al óptimo o no, por lo que la búsqueda local queda de la siguiente manera.

Algoritmo 3.3 Búsqueda local para el algoritmo GRASP.

Entrada: Matrices de datos, vector **rut[ ][ ]**,vector **rutbis[ ][ ]**, vector **val[ ]**

Salida: Las rutas que cada autobús tiene que seguir para recoger a los estudiantes.

```
1: loop = 0, flag = 0
2: mientras loop < loopmax hacer
3:    para  i = 0 to w, donde w es la cantidad de rutas existentes hacer
4:       Seleccionar dos elementos diferentes dentro de rutbis[i][ ] e intercambiar su posición
5:    fin para
6:    Calcular el valor de la nueva solución, donde el valor del recorrido de cada ruta se guarda
      en valbis[ ]
7:    para  i = 0 to w hacer
8:       si valbis[i] ≤ val[i]  entonces
9:          valbis[i] = val[i]
            rut[i][ ] =  rutbis[i][ ]
            Resetear el contador flag = 0
10:      si no
11:         Incrementar un contador flag++
12:      fin si
13:      si flag ≥ flagmax  entonces
14:         Seleccionar aleatoriamente dos rutas diferentes rutbis[i][ ] y rutbis[j][ ].
            Seleccionar aleatoriamente un elemento de cada ruta e intercabiarlos, siempre y
            cuando ambas rutas sigan cumpliendo la restricción de capacidad, si no la cumplen
            se eligen elementos diferentes hasta que se hace el intercambio.
15:      fin si
16:   fin para
17:   loop++
18: fin mientras
```

Debido a que en la búsqueda local estamos cambiando simultáneamente cada una de las rutas existentes, al momento de evaluar la solución generada debemos guardar el valor de cada ruta por separado, para que cuando comparemos una nueva solución contra la mejor que tenemos hasta el momento, la comparación sea ruta por ruta, ya que de lo contrario se corre el riesgo de pasar por alto alguna configuración de una ruta que pueda mejorar la solución aunque la suma de todas ellas sea peor que la que tenemos hasta el momento.

Una vez considerado este ajuste el algoritmo completo queda de la siguiente manera.

Algoritmo 3.4 GRASP para resolver el SBRP

Entrada: Matrices de datos.

Salida: Distancia total recorrida por los autobuses y los estudiantes, y los puntos que recorre cada ruta.

1: Inicializar **val_final** = M, donde M es un valor muy grande.
Inicializar las matrices **rut[ ][ ]**, **rutbis[ ][ ]** y **rutfinal[ ][ ]**, con ceros, en donde se guardaran los puntos que recorren cada ruta.
2: **mientras** $K < 10$ **hacer**
3: Crear una solución inicial factible para el problema, **SolIni();**, las rutas creadas se guardan en **rut[ ][ ]**.
4: Calcular el valor de la solución inicial, **val[ ]**.
5: Copiar las rutas a la matriz **rutbis[ ][ ]**.
rutbis[ ][ ] = **rut[ ][ ]**
6: Mejorar la solución mediante una búsqueda local, **BusLoc();**
7: **si val** $<$ **val_final entonces**
8: **val_final** = **val**
rutfinal[ ][ ] = **rutbis[ ][ ]**
9: **fin si**
10: $K++$
11: **fin mientras**
12: **imprimir val_final** y **rutfinal[ ][ ]**

Si bien con este método no garantizamos el obtener la solución óptima para el problema, si nos brinda una solución de buena calidad, en poco tiempo en comparación a lo que nos tardaríamos en encontrar la solución óptima,

3.5. Comparación de resultados

La formulación lineal entera se resolvió mediante el uso del software glpk y el algoritmo GRASP se implementó en C++ realizando 20 corridas para cada instancia y reportando el mejor resultado obtenido.

Si bien uno de los objetivos de este trabajo es la comparación entre el método exacto utilizado para resolver el problema contra el método heurístico, al igual que en el trabajo del 2013 de Kim y Park [10], no nos fue posible hacer una comparación a gran escala ya que el método exacto utilizado, si bien nos da una solución para instancias del problema, dichas instancias son muy pequeñas, por lo que no se puede hacer una comparación que sea significativa, como se muestra en el siguiente cuadro.

Instancia	Rutas	Puntos	M. Exacto	Tiempo(seg)	M. Heurístico	Tiempo(seg)
1	2	7	3365	2.3	3365	0.2
2	2	10	4427	3.2	4427	0.2
3	2	11	5681	5.7	5681	0.3
4	2	15	-	-	6673	0.3
5	3	15	-	-	8845	0.4
6	3	25	-	-	9314	0.51
7	4	40	-	-	17435	0.86
8	4	50	-	-	20786	1.9

Cuadro 3.2: Comparación entre los métodos implementados

Como se puede observar en las instancias pequeñas el GRASP muestra un desempeño bastante bueno y aunque no podemos saber con certeza qué tan buenas son las soluciones para las instancias grandes, suponemos que conforme se va incrementando el tamaño de las instancias, las soluciones tenderan a degenerarse, sin embargo, dichas soluciones se pueden considerar aceptables considerando el tiempo de ejecución requerido para obtenerlas.

Ya que no se puden comparar los métodos de la forma que se esperaba, intuitivamente se pensaría en comparar instancias del problema para probar la calidad del heurístico realizado, sin embargo, la versión del problema que se propone cuenta con elementos que otras versiones del problema no consideran, en especial la parte de asignación de los estudiantes, las restricciones supuestas y la función objetivo, por lo que al momento de comprar instancias que no consideran esta parte se pondría en desventaja el algoritmo utilizado. El siguiente cuadro muestra las diferencias entre las heurísticas que más similitud muestran con nuestra versión del problema.

Referencia	No. de Escuelas	Flotilla	Objetivos	Restricciones	Area
Russell y Morrel [15]	Multiples	HO	TBD	C, MRT	Tulsa, Oklahoma
Braca *et al.* [5]	Multiples	HO	N	C,MRT,TW, EPT,MSN	Manhattan, New York
Ripplinger [14]	Unica	HT	N, TBD	C, MRT	Artificial
Trabajo presentado	Unica	HO	TBD	C, N	Artificial

Cuadro 3.3: Características de los diferententes métodos heurísticos

HO: Homogéneo.
HT: Heterogéneo.
N: Número de autobuses usados.
TBD: Distancia (tiempo) total recorrida por los autobuses.

C: Capacidad del vehículo.
MRT: Tiempo maximo abordo del autobús.
TW: Ventanas de tiempo.
EPT: Menor tiempo en que se rocoge un estudiante.
MSN: Minimo número de estudiantes para crear una ruta.

Capítulo 4
Algoritmo de aproximación

Cuando se tiene un problema de optimización que no se puede resolver de forma exacta y las heurísticas que lo resuelven no nos dan una garantía de que alejados estamos del óptimo, surge la alternativa de utilizar algoritmos de aproximación los cuales trabajan en tiempo polinomial y nos dan una garantía de qué tan buena (o mala) va a ser la solución de éste. Para definir un algoritmo de aproximación, necesitamos definir primero qué es el desempeño.

Definición 4.1 *Sea $e \in I$ y $s \in S(e)$, donde I es un conjunto de instancias y $S(e)$ su conjunto de soluciones. Entonces el desempeño de s con respecto de e está dado por*

$$R(e, s) = \frac{m(e,s)}{OPT(e)}$$

es decir, la razón entre el valor de s y el valor de una solución óptima.

Note que $R(e, s) \leq 1$ es un problema de maximización y $R(e, s) \geq 1$ es un problema de minimización. Del concepto de desempeño podemos derivar el término garantía.

Definición 4.2 *Sea $OPT \in NPO$ de minimización, sea A un algoritmo que para toda $e \in I$ regresa $A(e) \in S(e)$ y sea $\alpha : \mathbb{N} \to \mathbb{R}^+$. Si A cumple que*

$$R(e, A(e)) \leq \alpha(|e|)$$

para toda $e \in I$, entonces diremos que A es un algoritmo de aproximación para OPT con una garantía de α. Si además A se ejecuta en tiempo polinomial, diremos que OPT se puede aproximar con una garantía α.

4.1. Antecedentes

Debido a la complejidad que supone resover el problema de forma general, Bock et. al.[3] consideran una estructura de árbol para el mismo, en el cual dejan de lado la parte de asignación, es decir, se sabe con exactitud la cantidad de estudiantes que hay en cada vértice desde el principio del problema, ademas introducen un parámetro el cual llaman "regret". Formalmente definen el problema como una gráfica no dirigida $G = (V, E)$ con distancia en las aristas $d : E \to \mathbb{Z}_+$, un nodo $s \in V$ que representa a la escuela y un conjunto $W \subseteq V$ que

representa la casa de los niños. Adicionalmente, se tiene la capacidad de los autobuses $C \in \mathbb{Z}_+$ y el parámetro de regret $R \in \mathbb{Z}_+$.

El objetivo es construir un conjunto de cardinalidad mínima $\mathcal{P}$ de caminatas que terminen en el vértice s (rutas de autobús), asignar cada estudiante $w \in W$ para que sea responsabilidad de un autobús $p(w) \in \mathcal{P}$, tal que para cada caminata $P \in \mathcal{P}$, el número total de niños w con $p(w) = P$ es a lo más la capacidad C; para cada niño el parámetro de regret se respeta, esto es: $d^P(w, s) \leq d(w, s) + R$, donde $d^P(w, s)$ es la distancia del niño w a la escuela s en la caminata $p(w)$, y $d(w, s)$ es la distancia más corta desde w hasta s en la gráfica G.

Dan un algoritmo de aproximación con un factor de aproximación de 4 para el problema en general sobre árboles y además para la versión del problema donde se consideran los regrets dan un algoritmo con factor de aproximación 12.5 de igual manera sobre árboles pero sin considerar la capacidad de los autobuses.

4.2. Algoritmo

Dada la poco literatura existente en lo que respecta a algoritmos de aproximación para el problema de ruteo del autobús escolar, se propone un algoritmo para resolver el problema de forma general, es decir, sin las consideraciones adicionales presentadas en el capitulo anterior. Ahora se va a considerar como función objetivo el minimizar la cantidad de autobuses necesarios para realizar la recolección, todos los autobuses tiene la misma capacidad la cual no puede ser superada y además existe una distancia máxima que pueden recorrer.

Tomando como base el trabajo realizado por Arkin et. al.[1] en el cual propone varios algoritmos de aproximación para diferentes variantes del problema de ruteo vehicular, trasladamos uno de sus resultados obtenidos a nuestro problema, para lo cual se sabe la cantidad de estudiantes que hay en cada punto de recolección, por lo que ahora definimos el problema de la siguiente manera.

Sea $G = (V, E)$ la gráfica completa asociada al problema y $G' = (V', E')$ la subgráfica inducida, que como se sabe es una subgráfica tal que E' son todas las aristas de G cuyos extremos están en V', definida por los vértices correspondiente a los puntos que se tiene que visitar (ya sea paraderos o casa de los estudiantes), los cuales deben ser visitados una sola vez por algún autobús. Estos vértices tienen asociado un valor $val(v_i) \geq 1$ el cual denota la cantidad de estudiantes existentes en ese punto, y cada arista $(v_i, v_j) \in E'$ tiene un peso $d_{ij} \geq 0$ que corresponde a la distancia que hay ente el punto v_i y el punto v_j.

A diferencia de los supuestos originales presentados para el problema, se considera que el número de autobuses a utilizar no ésta definido y se tratará de minimizar este número junto con la distancia que recorre cada autobús en su ruta. Se propone a continuación un algoritmo

de aproximación para el número mínimo de autobuses requerido.

El objetivo que persigue Arkin para el algoritmo que propone, consiste en minimizar las trayectorias utilizadas para cubrir todos los vértices de la gráfica, con la restricción adicional de que el valor de la distancia en cada trayectoria tiene que ser a lo más γ. Además de los supuestos originales de Arkin nosotros agregamos una restricción adicional con respecto a las trayectorias, donde ahora no solo se limita la distancia máxima de la misma, sino que ahora los vértices tienen un peso y la suma del peso de todos los vértices en una trayectoria tiene que ser a lo más λ, que es la capacidad que tienen los autobuses.

La forma en que funciona el algoritmo, consiste en que a cada paso se crea un bosque con n-k aristas como se muestra en la figura 4.1, lo cual va a formar k componentes conexas.

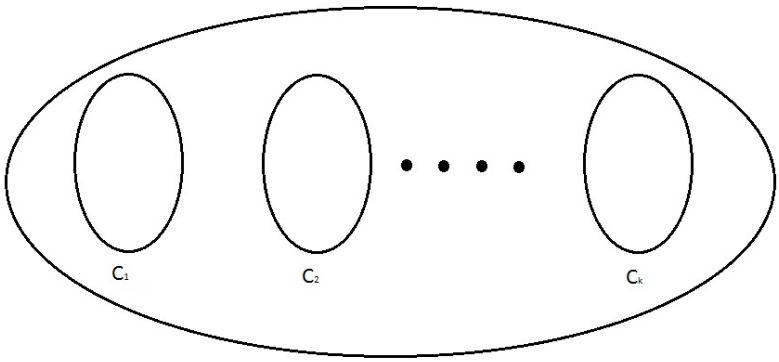

Figura 4.1: Ejemplo de el bosque que se forma en la iteración K.

Para cada componente conexa se forma una trayectoria donde sus vértices tengan grado a lo más dos, después se duplican las nuevas aristas obtenidas lo que nos forma un ciclo y finalmente eliminamos una de las aristas de mayor peso, lo que nos deja nuevamente con una trayectoria, como se muestra en la fígura 4.2, dicha trayectoria se descompone en subtrayectorias, donde cada subtrayectoria creada cumple las restricciones de distancia máxima, así como de capacidad y se lleva la cuenta de las subtrayectorias creadas para cada bosque con n-k aristas.

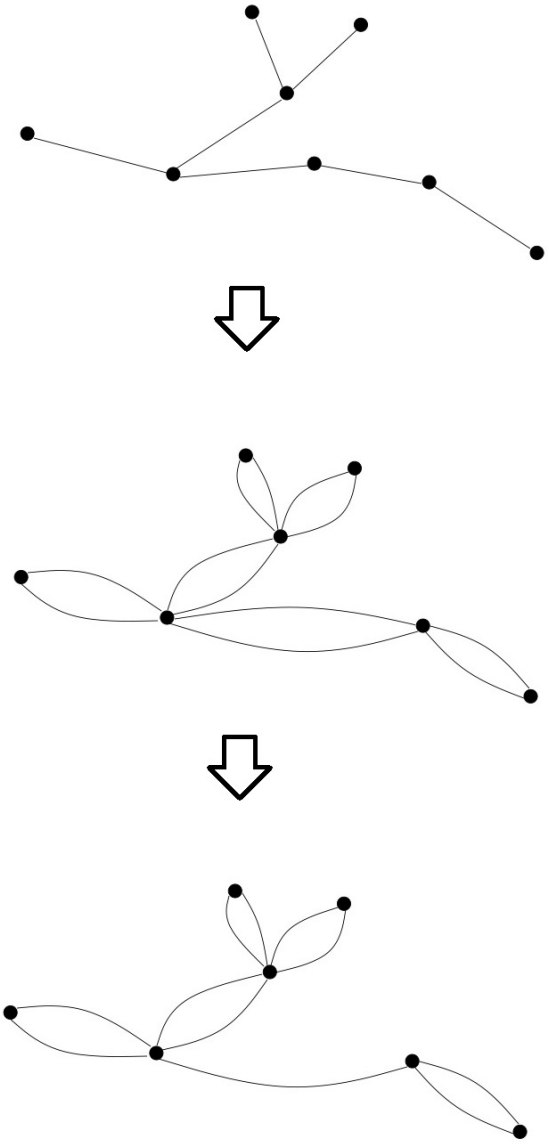

Figura 4.2: Pasos a seguir dentro de cada componente conexa.

Para la construcción del algoritmo debemos denotar a $N(P)$ como la cantidad de trayectorias generadas en el camino P. Para la primera parte del algoritmo, donde se busca crear un bosque de $n-k$ aristas no se realizan modificaciones, ya que la forma de cómo se construye el bosque no cambia. La segunda parte del algoritmo donde se forman las trayectorias, ahora el v_j no solo cumple que la distancia de v_i hasta v_j sea menor que γ, si no que además ahora debe cumplir $\sum_i^j val(v_l) \leq \lambda$. Adicionalmente cada componente conexa $'i'$ que se forma cumple con $\delta(TOUR_i) \leq \lceil \frac{\zeta(TOUR_i)}{2\gamma} \rceil \lambda$, por lo que el algoritmo ahora queda de la siguiente manera.

Algoritmo 4.1 SBRP k paths

Entrada: La gráfica completa $G' = (V', E')$
Una métrica $\zeta : V'xV' \to \mathbb{R}_+$
Un número real $\lambda > 0$
Un número real $\gamma > 0$

Salida: Un conjunto de trayectorias, la suma de los elementos de cada trayectoria tiene un valor de a lo más de λ y cada trayectoria tiene una longitud de a lo más γ que cubren V'.

1: **para** $k = 1, 2, \ldots, n$. |V'|=n **hacer**
2: $F =$ un bosque de peso mínimo en G' con $n - k$ aristas.
$C_1, C_2, \ldots, C_k =$ las componentes conexas de F
$SOL_k = \emptyset$
3: **para** i=1,2,...,k **hacer**
4: $N(i, k) = 1$
$TOUR_i =$ un tour con los vértices de C_i
$P_i =$ una trayectoria que resulta de eliminar una arista de $TOUR_i$.
5: **mientras** $\delta(P_i) > \lambda$ y $\zeta(P_i) > \gamma$ **hacer**
6: $(v_1^i, v_2^i, \ldots, v_{|P_i|}^i) = P_i$
$j =$ el índice del primer vértice en P_i tal que la longitud de la $v_1^i - v_{j+1}^i$ subtrayectoria de P_i es mayor que γ o la suma de los vértices es mayor que λ.
$P_i = P_i \setminus \{(v_1^i, v_2^i), (v_2^i, v_3^i), \ldots, (v_{j-1}^i, v_j^i), (v_j^i, v_{j+1}^i)\}$
$SOL_k = SOL_k \cup \{v_1^i, (v_1^i, v_2^i), v_2^i, (v_2^i, v_3^i), \ldots, (v_{j-1}^i, v_j^i), v_j^i\}$
$N(i, k) = N(i, k) + 1$
7: **fin mientras**$SOL_k = SOL_k \cup P_i$
8: **fin para**
9: **fin para**
10: $\hat{k} = argmin_k \sum_{i=1}^{k} N(i, k)$
11: **devolver** $SOL_{\hat{k}}$

Si bien la modificación realizada nos ayuda a acotar el peso máximo que tienen los vértices en cada trayectoria, esta restricción no modifica las propiedades bajo lo que asume Arkin et.al., por lo que la demostración sigue siendo la misma que ellos presentan.

Teorema 4.1 *El algoritmo SBRP k paths, es un algoritmo de aproximación con un factor de 3 que corre en tiempo polinomial.*

Demostracion 1 *El algoritmo corre en tiempo polinomial ya que el bosque F puede ser calculado en tiempo polinomial (simplemente deteniendo el algoritmo glotón para calcular el árbol de peso minimo, después de escoger las primeras $n - k$ aristas), y los demás pasos se pueden*

realizar claramente en tiempo polinomial.

El algoritmo prueba todas las posibilidades de k y escoge la mejor. Consideremos la iteración en donde $k = k^*$.

La solución óptima está compuesta por k trayectorias, donde cada trayectoria tiene una longitud de a lo mas γ y la suma del valor de sus vértices es a lo más λ. Por lo que, $\zeta(OPT) \leq \gamma k$. Esta solución óptima tiene $n - k$ aristas, y dado que F es un bosque de costo mínimo con $n - k$ aristas, , $\zeta(F) \leq \gamma k$, esto implica que $\sum_{i=1}^{k} \zeta(TOUR_i) \leq 2\gamma k$.

Para toda i se satisface lo siguiente:

$$N(i,k) \leq \lceil \frac{\zeta(TOUR_i)}{\gamma} \rceil \leq \frac{\zeta(TOUR_i)}{\gamma} + 1$$

Por lo que

$$|SOL_k| = \sum_{i=1}^{k} N(i,k) \leq \frac{1}{\gamma} \sum_{i=1}^{k} \zeta(TOUR_i) + k \leq \frac{2\gamma k}{\gamma} + k = 3k$$

y el algoritmo regresa una solución de a lo más $3k^*$ trayectorias.

Una vez que se tiene las k trayectorias, sólo queda conectarlas de tal manera que todas comiencen en el vértice S_0 y terminen en S_1.

A diferencia del algoritmo heurístico presentado en la sección anterior, la solución del algoritmo de aproximación nos da un parámetro el cual indica qué tan alejada está la solución encontrada de la solución óptima.

Capítulo 5
Conclusiones y trabajo futuro

Este trabajo comenzó con el estudio de una nueva versión del problema de ruteo del aubús escolar, la cual es significativamente más difícil debido a la parte de asignación que se resuelve dentro del problema y a la que denotamos como con recolección mixta, para el cual se pretende encontrar la solución para algunas de sus intancias, ya sea de manera óptima o aproximada.

Al ser un problema NP-duro, es difícil diseñar un modelo mediante el cual se pueda obtener una solución exacta para el problema incluso para versiones relajadas en comparación al problema propuesto, fue por ello que se utilizaron diferentes herramientas para encontrar una solución al problema propuesto, empezando por un método exacto el cual en la literatura existe casi siempre es descartado para dar una solución, ya que la configuración del problema hace muy difícil que se pueda dar una solución considerando que la mayoría de las instancias de los mismos se basan en casos reales, por lo que el número de estudiantes y paradas de autobuses es bastante grande.

Nos fue posible diseñar una formulación matemática lineal que puede fácilmente transformarse a un programa lineal entero y obtener una solución exacta para algunas instancias del problema, por desgracia el tamaño de las instancias las cuales pueden ser resueltas por este método son muy pequeñas. La formulación por sí misma abre la posibilidad a que se puedan resolver variantes del problema como:

- Cada parada de autobús pueda ser visitada por más de un autobús, esto brindaría flexiblidad al momento de crear las rutas.
- Utilizar ventanas de tiempo junto con vehículos con diferentes capacidades.
- Junto con el punto anterior considerar el caso donde el servicio es para más de una escuela.

A futuro se puede pensar en hacer una revisión de la formulación propuesta con el propósito de detectar alguna restricción o variable que no sea necesaria y de esta forma decrementar ya sea la cantidad de restricciones o las variables utilizadas, para que sea posible resolver

instancias más grandes del problema.

Para la parte heurística, de la cual existe una extensa literatura al respecto, donde se explota por completo todo su potencial, por lo que se enfocan en resolver versiones multiobjetivo del problema y ya que nuestra versión del problema considera la asignación de los estudiantes entre otras cosas, se decidió no manejar una versión multiobjetivo del problema y utilizar un algoritmo clásico para encontrar una solución para el mismo.

La elección del GRASP nos dio la oportunidad de encontrar una solución al problema en un tiempo razonable, sin embargo se depende de que en la solución inicial creada se escojan las asignaciones de manera correcta para poder llegar al óptimo. Dentro del proceso mejora de la solución se evitó hacer cambios tanto a la asignación de estudiantes como a la cantidad de elementos en cada ruta, si bien esto puede sonar a que se limita la búsqueda en el espacio de soluciones, dada la naturaleza del problema es muy difícil determinar un momento en el cual se necesite realizar un cambio de esta naturaleza para mejorar la solución.

Sin embargo, la propia naturaleza del GRASP nos permite explorar varias soluciones distintas con una sola ejecución del algoritmo, si a esto le sumamos que el algoritmo se ejecuta varias veces por ser un algoritmo multiarranque, esto nos permite explorar diferentes configuraciones de la solución inicial sin tener que realizar cambios en ella a la mitad del proceso de mejora.

- Implementar una búsqueda local la cual nos permite encontrar la solución óptima sin importar la configuración de la solución inicial.
- Diseñar una metaheurística para encontrar una solución para el problema.

Finalmente en cuanto a la parte del algoritmo de aproximación dejamos de lado la versión del problema que propusimos para dar un algoritmo de aproximación para el problema de forma general y de esta forma contribuir con la escasa literatura existente acerca de algoritmos de aproximación para el problema de ruteo del autobús escolar, a futuro se puede considerar:

- Proponer un algoritmo de aproximación el cual considere la versión del problema propuesto en el capitulo 3.
- Estudiar estructuras o familias de gráficas en las cuales se pueda ser mas sencillo dar un algoritmo de aproximación para el problema original.

La utilización de todos estos métodos para encontrar una solución al problema nos permitió observar el alcance que tiene cada una de estas técnicas para problemas de ruteo tan complejos como el que fue presentado.

Bibliografía

[1] Arkin, E., Hassin, R., and Levin, A., (2006), *Approximation algorithms for minimum and min-max vehicle routing.* Journal of Algorithms 59 , 1-18.

[2] Bektas, T., Elmastas, S., (2007) *Solving school bus routing problems through integer programming.* Journal of the Operational Research Society 58 (12), 1599-1604.

[3] Bock, A., Grant, E., Könemann, J., and Sanità. L.,(2012) *The school bus problem in trees.* Algorithmica 67, pp 49-64.

[4] Bowerman, R., B. Hall, and P. Calamai (1995). *A Multi-Objective Optimization Approach to Urban School Bus Routing: Formulation and Solution Method*. Transportation Research A 29A, pp. 107-123.

[5] Braca, J., J. Bramel, B. Posner and D. Simchi-Levi (1997). *A Computerized Approach to the New York City School Bus Routing Problem.* HE Transactions 29, pp. 693-702.

[6] Clarke, G. and J. W. Wright (1964). *Scheduling of vehicles from a central depot to a number of delivery points*. Operations Research 12, pp. 568- 581.

[7] Dantzig, G.B. and J.H. Ramser (1959). *The truck dispatching problem*. Management Science 6, pp. 81-91.

[8] Davis, N., et. al., (2004) *Mexico City Vehicle Activity Study*. from ISSRC Webpage: http://www.issrc.org/ive/downloads/reports/MexicoCityMexico.pdf

[9] Laporte, G. and Y. Nobert (1987). *Exact algorithms for the vehicle routing problem.* Annals of Discrete Mathematics 31, pp. 147-184.

[10] Kim, T., and Park, B.J., (2013). *Model and Algorithm for Solving School Bus Problem.* Journal of Emerging Trends in Computing and Information Sciences, 596-600.

[11] Li, L.Y.O. and Z. Fu (2002). *The school bus routing problem: a case study.* Journal of the Operational Research Society 53, pp. 552-558

[12] Newton, R. M. and W. H. Thomas (1969). *Design of School Bus Routes by Computer.* Socio-Economic Planning Science 3, pp. 75-85.

[13] Newton, R. M. and W. H. Thomas (1974). *Bus routing in a multi-school system.* Computers and Operations Research 1, pp. 213-222.

[14] Ripplinger, D., (2005). *Rural school vehicle routing problem.* Transportation Research Record 1992, 105-110.

[15] Russell, R.A., Morrel, R.B., (1986). *Routing special-education school buses.* Interfaces 16 (5), 56-64.